

This paper is a preprint (IEEE “accepted” status).

IEEE copyright notice. © 2011 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

DOI. 10.1109/CCP.2011.22

<http://doi.ieeecomputersociety.org/10.1109/CCP.2011.22>

Combining non-stationary prediction, optimization and mixing for data compression

Christopher Mattern
Fakultät für Informatik und Automatisierung
Technische Universität Ilmenau
Ilmenau, Germany
christopher.mattern@tu-ilmenau.de

Abstract—In this paper an approach to modelling non-stationary binary sequences, i.e., predicting the probability of upcoming symbols, is presented. After studying the prediction model we evaluate its performance in two non-artificial test cases. First the model is compared to the Laplace and Krichevsky-Trofimov estimators. Secondly a statistical ensemble model for compressing Burrows-Wheeler-Transform output is worked out and evaluated. A systematic approach to the parameter optimization of an individual model and the ensemble model is stated.

Index Terms—data compression; sequential prediction; parameter optimization; numerical optimization; combining models; mixing; ensemble prediction

I. INTRODUCTION

A. Background

Sequential bitwise processing plays a key role in several general-purpose lossless data compression algorithms, including Dynamic Markov Coding (DMC) [1], Context Tree Weighting (CTW) [2] and the recently emerging “Pack” (PAQ) [3], [4] family of compression algorithms. All of these algorithms belong to the class of statistical data compression algorithms, which split the compression phase into modelling and coding. A statistical model assigns probabilities to upcoming symbols and these are translated into corresponding codes. Assigning a high probability to the actually upcoming symbol leads to a short encoding, thus producing compression. The ideal code length corresponding to a prediction can closely be approximated via Arithmetic Coding (AC) [5]. Hence improving prediction accuracy is crucial for compression.

Recently, PAQ-based compression algorithms have been of high public interest, due to the enormous compression achieved. Unfortunately, there is little up-to-date literature on the internals of the involved algorithms [3], [4], [6]. PAQ compression algorithms combine multiple binary predictors and are characterized by low processing speed and the best compression rates in multiple benchmarks up to date. Ensemble prediction has previously been applied successfully in other areas of research, e.g., time series forecast and classification [7], [8], [9], and form a promising direction of research. In the field of compression an ensemble approach is often called Context Mixing (CM).

The most elementary task of the prediction model is sequential probability assignment, i.e., predicting the probability distribution $P(y_{k+1}|y_k y_{k-1} \dots y_1)$ of the upcoming symbol

y_{k+1} based on the already encountered sequence $y_1 y_2 \dots y_k$ over a finite alphabet Σ . Such a task typically arises when working with context models. A finite number of symbols preceding y_{k+1} can be used to condition the probability, which leads to finite context modelling [5]. The finite context, e.g., the character immediately preceding the current one, splits the source sequence into sub-sequences. These are often called context histories. For instance the context history of the context “e” (underlined) regarding the last sentence is “sndxs”, an underscore represents a space symbol. This work focuses on binary alphabets, $\Sigma = \{0, 1\}$ and uses the convention $p_k = P(y_k = 1)$.

B. Previous work

Experiments have shown that a local adaption of the computed statistics during modelling typically improves compression. Thus more recent observations are of higher importance for probability assignment [5], [10]. This observation was made more or less accidentally due to limited calculation precision, which lead to a periodic rescaling of character counts [5]. Previous work investigated the effect of scaling [11] and pointed out an approximate probability estimation model for binary sequences based on exponential smoothing [10]. Another aspect is the presence of noise within observations. An imperfect choice of conditioning contexts will lead to observations within context histories, which deviate from the governing probability distribution. We consider such events as outliers or simply noise. A recent work [12] studied the effect of a limited probability interval, i.e., $\theta \in [\alpha, \beta] \subset [0, 1]$ along with the estimation of the parameter $p_k = \theta = \text{const}$ regarding a series of independent identically distributed (iid) binary random variables. A limited probability interval can be explained by viewing an observed sequence as the outcome of the transmission of the “true” sequence through a noisy channel (i.e., an extension to the original source model). Results indicate that having knowledge about the parameters α and β can lead to significant improvements in compression for short to medium sized sequences. Thus using the restriction $p_k \in [\alpha, \beta]$ can represent a countermeasure for noisy observations.

C. Our contribution

The previous section explained the aspects of observation recency and observation uncertainty. Based on these ideas we enhance a standard approach for sequential binary prediction and introduce a new prediction model. We further employ our prediction model to construct a new ensemble compression algorithm. This compression algorithm is intended to be used as a second step algorithm in Burrows-Wheeler-Transform (BWT) based compression. Both, the sequential prediction model and the ensemble compression algorithm, contain constants (fixed during compression or decompression), which influence the probability estimation and the compression. We denote such constants as parameters of the algorithm or parameters of the prediction model (which should not be confused with parameters of a distribution). In the general setting there are no simple rules for choosing the (unknown) parameters. Among the set of feasible parameters, we want to choose the parameters according to a certain objective. In data compression this objective is the minimization of the size of the compressed output. Most of the parameter optimization in the area of data compression was carried out using ad-hoc hand-tuning, e.g., [13, p. 4], [14, p. 6] and [15, p. 4]. In this work we want to introduce systematic approaches to automated parameter optimization, since these will improve the compression performance compared to ad-hoc hand-tuning.

We distinguish two versions of automatic parameter optimization in compression, which we call offline and online optimization. Given a training data set the models' parameters can be fitted once and remain static during future usage (offline optimization). This approach requires a carefully chosen set of training data. Since the optimization takes place only once and not prior to every compression pass there are no significant restrictions on the amount of data and the associated processing time. On the other hand, adding an initial optimization pass prior to compression and saving the parameters along with the compressed data refers to an online approach. However, there are more severe restrictions on the utilized resources. We consider a situation in which the optimization pass requires orders of magnitude more time than the actual (de-)compression process impractical for online optimization. In this work we focus on online optimization and incorporate an automated optimization pass into the ensemble model mentioned above. Coupling online optimization and statistical compression leads to asymmetric statistical compression, a new family of statistical compression algorithms. Without optimization such algorithms are typically symmetric, since modelling and coding is required during compression and decompression. Similar approaches to asymmetric algorithms exist in the field of audio compression [16].

There is another non-obvious benefit in using optimization. Assume an algorithm A achieves a certain compression rate using an ad-hoc parametrization. A computationally cheaper algorithm B produces compression comparable to A along with optimized parameters. Thus the compression time is

reduced when A is replaced by B . This argument holds especially for offline optimization: The time required for optimization does not need to be included in the compression time, since optimization is only carried out once.

The remaining part of this work is divided into four further sections. First we present a new elementary, binary prediction model, its application to non-binary alphabets and an approach to ensemble prediction. Section III briefly summarizes iterative numeric optimization and its application to the presented modelling algorithms. Afterwards Section IV evaluates the model components' performance and the impact of optimization.

II. MODELLING

A. Elementary prediction

As previously mentioned in Section I the most essential task is to estimate the probability distribution $p_{n+1} = P(Y_{n+1} = 1 | B_n = b_n)$ given the series of binary random variables $B_n = Y_1 Y_2 \dots Y_n$ and an instance $b_n = y_1 y_2 \dots y_n \in \{0, 1\}^n$, where m out of n bits are one. Assuming iid random variables Y_k , i.e., $p_k = \theta$ for all k and some fixed $\theta \in [0, 1]$, one can calculate the probability of a given outcome b_n via

$$P(B_n = b_n | \theta) = \prod_{k=1}^n P(Y_k = y_k) = \theta^m (1 - \theta)^{n-m}. \quad (1)$$

When b_n is fixed an estimation $\hat{\theta}$ of θ can be obtained via maximizing $P(\theta | b_n)$, or via minimizing the entropy

$$\begin{aligned} H(\theta | b_n) &= - \sum_{k=1}^n \log P(Y_k = y_k) \\ &= -m \log \theta + (n - m) \log(1 - \theta). \end{aligned} \quad (2)$$

Note that logarithms are to the base two. The result of minimizing (2) is the well-known maximum likelihood estimator $\hat{\theta} = m/n$. Equation (2) is rewritten to yield

$$H(b_n) = - \sum_{k=1}^n (y_k \log \theta + (1 - y_k) \log(1 - \theta)). \quad (3)$$

Since we assume that the coding cost of more recent events is of higher importance, we modify (3) to become a weighted entropy (cf. [11])

$$H_w(b_n) = - \sum_{k=1}^n c_k (y_k \log \theta + (1 - y_k) \log(1 - \theta)), \quad (4)$$

where $0 < c_1 < c_2 < \dots < c_n$ is some weight sequence. In this way, the value of θ is strongly linked to more recent observations (steps $n, n-1, \dots$).

Next we address the aspect of observation uncertainty, similar to [12]. The observations y_k are viewed to be the outcome of a binary symmetric channel. On the transmitter side the outcome y_k of a binary random variable Y_k is sent through the channel. The receiver observes a corrupted bit

$1 - y_k$ with a probability ε , i.e., the outcome of a binary random variable X_k . Summarizing

$$\begin{aligned} P(X_k = y_k | Y_k = y_k) &= 1 - \varepsilon, \\ P(X_k \neq y_k | Y_k = y_k) &= \varepsilon \end{aligned} \quad (5)$$

holds for some $0 \leq \varepsilon \leq 0.5$. Thus (4) is modified to become the expected, weighted entropy

$$\begin{aligned} \bar{H}_w(b_n) &= - \sum_{k=1}^n c_k (\delta_k \log \theta + (1 - \delta_k) \log(1 - \theta)), \\ \delta_k &= (1 - \varepsilon)y_k + \varepsilon(1 - y_k), \end{aligned} \quad (6)$$

since we can only observe the receiver side. We assume that the statistical properties of the bit sequence do not change rapidly (i.e., $p_{n+1} \approx p_n$) and approximate p_n using the solution of the minimum-entropy problem

$$p_{n+1} \approx p_n = \arg \min_{\theta} \bar{H}_w(b_n), \quad (7)$$

which results in

$$\begin{aligned} p_{n+1} &\approx \frac{\sum_{k=1}^n c_k \delta_k}{\sum_{k=1}^n c_k} \\ &= \varepsilon + (1 - 2\varepsilon) \frac{\sum_{k=1}^n c_k y_k}{\sum_{k=1}^n c_k}. \end{aligned} \quad (8)$$

Thus modelling uncertainty via (5) restricts the probability interval to be $[\varepsilon, 1 - \varepsilon]$. As a side effect the problem of assigning a probability to the opposite bit $y_{k+1} = 1 - b$ when processing a deterministic sequence $y_1 = y_2 = \dots = y_k = b \in \{0, 1\}$ is solved. The source model discussed above contains several (generally unknown) parameters - the weight sequence and ε . In order to use the source model for prediction these parameters have to be chosen. A bad choice leads to redundancy during coding, e.g., in some step k the actual value of p_k could be located outside of the restricted probability interval, but the model is only able to assign values in $[\hat{\varepsilon}, 1 - \hat{\varepsilon}]$ depending on the estimated parameter $\hat{\varepsilon}$.

B. Efficient approximations

Equation (8) can already be utilized to obtain a probability estimation given a weight sequence and ε . However, from a practical point of view and as a matter of convenience an estimation should be calculated incrementally, hence we select an exponentially decaying weight sequence

$$c_k = \lambda^{n-k}, \quad 1 \leq k \leq n, \quad (9)$$

with $\lambda \in (0, 1]$. Equation (8) becomes

$$p_{n+1} = \frac{S_{n+1}}{T_{n+1}}, \quad (10)$$

where

$$S_{n+1} = \lambda S_n + \delta_n, \quad (11)$$

$$T_{n+1} = \lambda T_n + 1, \quad (12)$$

Bit	s_1 72 (H)	s_2 101 (e)	s_3 108 (l)	s_4 108 (l)	...
y_8	01001000	<u>01100101</u>	01101100	01101100	...
y_7	01001000	<u>01100101</u>	<u>01101100</u>	01101100	...
...					
y_1	01001000	<u>01100101</u>	<u>01101100</u>	01101100	...
y_8	01001000	01100101	<u>01101100</u>	01101100	...
y_7	01001000	01100101	<u>01101100</u>	<u>01101100</u>	...
...					

Fig. 1. The decomposed symbol s_3 is encoded in eight consecutive binary steps (current bit is boldface) using an order-1 context (underlined), i.e., the predictions $P(y_8 = 1 | s_2 = 101)$, $P(y_7 = 1 | y_8 = 1, s_2 = 101)$, ..., $P(y_1 = 1 | y_2 y_3 \dots y_8 = 0110110, s_2 = 101)$ need to be calculated. After encoding s_3 , s_4 can be processed in the same fashion.

which can be reformulated to yield an adjustment proportional to the prediction error

$$p_{n+1} = p_n + \frac{1}{T_{n+1}} (\delta_n - p_n). \quad (13)$$

Initially we have $p_0 = 0.5$ and $T_0 = 0$. Note that the sequence T_n is a geometric series and therefore

$$T_n \xrightarrow{n \rightarrow \infty} \frac{1}{1 - \lambda}. \quad (14)$$

For a very long sequence exponential smoothing can be used as an approximation of (13), i.e.,

$$\begin{aligned} p_{n+1} &= p_n + (1 - \lambda)(\delta_n - p_n), \\ &= \lambda p_n + (1 - \lambda)\delta_n. \end{aligned} \quad (15)$$

Depending on the computational resources different approximations seem acceptable:

- **Exact model M_1 .** An estimator state is (p_n, T_n) , computed according to (13).
- **Exponential smoothing M_2 .** The state is given by (p_n) and is updated following (15). Selecting $1 - \lambda = 2^{-l}$, $l \in \mathbb{N}$ results in a very efficient calculation using bit shifts and additions/subtractions only.

Note that M_1 can be approximated more closely by imposing an upper limit on T_n , or n , respectively. This yields a state (p_n, n') , with $n' = \min(n, \bar{n})$ for a threshold \bar{n} . The values of $1/T_n$ are found using a lookup table and $T_{\bar{n}} \approx T_{\infty}$ is set according to (14). All approximations described above share the same parameters λ and ε .

C. Alphabet decomposition and context modelling

The previous section dealt with the modelling of a binary alphabet. In general the compression algorithms work on n -ary alphabets Σ , typically $|\Sigma| = 2^8$. Hence bitwise processing requires an alphabet decomposition, i.e., a mapping $\text{code} : \Sigma \rightarrow \{0, 1\}^+$ and $\text{len} : \Sigma \rightarrow \mathbb{N}$ to indicate the code length. Without loss of generality we may assume that $\Sigma = \{0, 1, 2, \dots, m - 1\}$. Within this work we use a fixed decomposition, which we call “flat decomposition”, i.e., $\text{code}(s) = \text{bin}(s)$ (e.g., $\text{code}(65) = 01000001$) and $\text{len}(s) = L = 8$ for every symbol $s \in \Sigma$. Modelling

the probability distribution of s is split into $\text{len}(s) = L$ consecutive steps

$$P(s) = P(y_L)P(y_{L-1}|y_L) \dots P(y_1|y_2y_3 \dots y_L). \quad (16)$$

Working with conditional probabilities increases the prediction accuracy. A natural choice are order- N contexts, which have successfully been applied to text compression [4], [5]. An order- N context consists of the last N characters immediately preceding the current one. Figure 1 illustrates the bitwise modelling process using an order-1 context. Depending on the underlying data other choices can be reasonable as well, e.g., the neighbouring pixels in image compression [17], [18].

D. An ensemble predictor

Section I mentioned the successful application of ensemble models in other areas. In the area of compression such techniques are known [5], but there has been less interest in directly applying them. Such techniques allow multiple models to contribute with their advantages without cumulating their disadvantages [6]. During modelling a probability must be calculated for each alphabet symbol $s \in \Sigma$, hence combining M models roughly requires $M \cdot |\Sigma|$ operations. On the other hand bitwise processing just requires $M \cdot \bar{L}$ operations on average, where \bar{L} is the average code length. Without making further assumptions about symbol frequencies, i.e., applying the decomposition described in Section II-C, we get $\bar{L} = L = \lceil \log |\Sigma| \rceil$. An advantage of CM compared to Prediction by Partial Matching (PPM) is that it does not need to handle symbols, which did not appear in the current context, in a special way [19]. PPM indicates the presence of such a situation using an artificial escape symbol, whose probability needs to be modelled in every context. However, such situations may add redundancy, since there is code space allocated for possibly never appearing symbols. This issue can be crucial for PPM [20]. A disadvantage of CM is the requirement of multiple models simultaneously, which has heavy impact on processing speed and memory requirements.

We now describe the outline of our approach to ensemble prediction, or CM respectively. It is based on a source switching model [6]. Consider a set of M sources and a probabilistic switching mechanism, which selects source i with a probability of w_k^i (in step k) where $\sum_{i=1}^M w_k^i = 1$. Note that the switching model should not be confused with Volf's switching method [15], which is based on switching between *source coding algorithms* rather than constructing an ensemble source model. In its current state x_k^i the selected source emits a one-bit with the probability $p_k^i = P(Y_k = 1|x_k^i)$. Afterwards a state transition takes place for each source resulting in the next state x_{k+1}^i . In an analogous fashion the switching probabilities may vary, i.e., these may depend on a state, too. Summarizing the probability of a one-bit in step k is

$$p_k = \sum_{i=1}^M w_k^i p_k^i. \quad (17)$$

Thus the assumption (or approximation) of a switching source results in a linear ensemble prediction (linear mixing). Unfor-

```
eiehdnkleeeeeeeeeeeeeiiiiiiiiiiiiiyyeieei
ieeeeeieeeeeieeeeeeeeeeeeeyiyiyiii
iiyyiyiyiyiyiyiyiyiyiyiyiyieeeeeeeee
eeeeeeeeeeeeeeeeyeeeeeeceeeeeeeeeieeee
eeeeehhohhhhheeeeeeeeeeeeeeeeeieeeeeee
eeeeeeeeeeeeeeeeyeeeeeeeeeeeeeeeeeeee
iiiiiiiiiiiiiiiiiiiiiiiiiiiihheeeeeee
eeeeeeeeeeeeeeeeieeeeeeeeeeeeeeeeeyee
eeeeeeeeeeeeyeeeeeeeeeeeeeeeeeeeeeeee
eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
```

Fig. 2. A typical example of BWT output taken from *book1* (Calgary Corpus).

tunately, normally no information about the internals of the source (e.g., involved states and transitions) or the characteristics of the probability assignment is available. The assignment is up to the designer.

E. Applications and test cases

Single model: To examine the prediction model described in Sections II-A and II-B we will compare its performance to the well-known Laplace (LP)- and Krichevsky-Trofimov (KT)-estimators [2], [21] with scaling [11].

Ensemble model: For testing the ensemble approach we introduce a simple ensemble compression algorithm intended as a second step algorithm in BWT based compression. BWT sorts the characters in its input by context, hence it groups similar contexts together [22]. Since these contexts are often succeeded by the same characters BWT output mostly consists of long interleaved runs of characters, see Fig. 2. Such sequences can be modelled as non-stationary [13]. We model the BWT output as the outcome of a switching source, which consists of two individual non-stationary sources. One source randomly emits characters independent of the previous sequence (order-0), this is intended to model interruptions in a single characters run. A second source emits characters based on the character immediately preceding the current position (order-1). In contrast to the first source it is intended to model the long runs of identical characters. The individual models are implemented using the binary predictors described in Section II. We assume the switching probabilities to be constant, i.e., $w_k^2 = 1 - w_k^1 = \omega \in [0, 1]$.

Each individual model presented in Section II-B has two parameters λ and ε . The previously described BWT postprocessor has five parameters, $\lambda_1, \varepsilon_1, \lambda_2, \varepsilon_2$ and ω , respectively. Following these observations the next section will provide a way of optimizing the parameters.

III. OPTIMIZATION

A. Iterative numeric optimization

We decompose a model into its structure and parameters. Improving the model structure is a task which is typically carried out by humans. Model parameters can be fitted automatically to a typical training data set. There are different approaches, depending on the optimization target. A differentiable optimization target allows the usage of local search procedures, for instance Newton's Method, see standard

$\mathbf{x}_0 \leftarrow$ initial estimation
 $k \leftarrow 0$
repeat
 compute a search direction \mathbf{d}_k along which f decreases
 perform a line search $\alpha_k \leftarrow \arg \min_{\alpha} f(\mathbf{x}_k + \alpha \mathbf{d}_k)$
 update the solution $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{d}_k$
 next step $k \leftarrow k + 1$
until stopping condition met

Fig. 3. Basic outline of an iterative numeric minimization algorithm.

literature on these well-known techniques, e.g., [23]. When no derivative information is available (i.e., a non-differentiable optimization target) or the search space is highly multimodal other stochastic search techniques should be preferred, see e.g., [24]. In our setting we want to minimize the average code length f , depending on the parameters \mathbf{x} of the prediction model

$$\min_{\mathbf{x} \leq \underline{\mathbf{x}} \leq \bar{\mathbf{x}}} f(\mathbf{x}), \quad (18)$$

where $f(\mathbf{x})$ is given by a modification of (3)

$$f(\mathbf{x}) = -\frac{1}{n} \sum_{k=1}^n (y_k \log p_k(\mathbf{x}) + (1 - y_k) \log(1 - p_k(\mathbf{x}))). \quad (19)$$

Here boldface symbols indicate matrices or vectors. The parameter search should take place within the hypercube formed by the inequality constraints $\mathbf{x} \in [\underline{\mathbf{x}}, \bar{\mathbf{x}}] \subset \mathbb{R}^N$. In this work we want to focus on derivative-based optimization techniques based on Quadratic Programming, since f is differentiable. Figure 3 shows the typical outline of such an optimization procedure. The models described in the previous Section span a low-dimensional search space, e.g., $\mathbf{x} = (\lambda_1, \varepsilon_1, \lambda_2, \varepsilon_2, \omega)^T \in \mathbb{R}^5$. Opposed to the small number of parameters a function evaluation is, depending on the amount of training data, time consuming. It requires to run the corresponding model along with the calculation of derivatives. Since we want to use an online-optimization approach, the “training data” is the data to be actually compressed, i.e., we know it prior to optimization.

B. Estimating the search direction

Consider a quadratic approximation $f(\mathbf{x}_k + \mathbf{d}_k)$ of the target function f as a result of the Taylor-expansion at \mathbf{x}_k

$$f(\mathbf{x}_k + \mathbf{d}_k) \approx f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T \mathbf{d}_k + \frac{1}{2} \mathbf{d}_k^T \nabla^2 f(\mathbf{x}_k) \mathbf{d}_k. \quad (20)$$

Differentiating (20) in \mathbf{d}_k and solving for its roots yields a search direction

$$\mathbf{d}_k = \underbrace{-\nabla^2 f(\mathbf{x}_k)^{-1}}_{\mathbf{S}_k} \nabla f(\mathbf{x}_k). \quad (21)$$

The matrix \mathbf{S}_k can either be estimated iteratively or computed directly. Given a valid point $\mathbf{x}_k \in [\underline{\mathbf{x}}, \bar{\mathbf{x}}]$ a step towards \mathbf{d}_k might lead to a violation of the constraints. Hence the constraints influence the computation of \mathbf{d}_k . In order to calculate a feasible direction we adopt a slight modification of the method in [25], which we will now summarize briefly.

First the index set of binding constraints

$$I_k = \underbrace{I'_k(-\nabla f(\mathbf{x}_k))}_{I_k^1} \cup \underbrace{I'_k(-\mathbf{S}'_k \nabla f(\mathbf{x}_k))}_{I_k^2} \quad (22)$$

is identified depending on

$$I'_k(\delta) = \{i \mid x_k^i = \underline{x}_i \wedge \delta_i < 0 \vee x_k^i = \bar{x}_i \wedge \delta_i > 0\}. \quad (23)$$

An element $s_k^{ij'}$ of \mathbf{S}'_k is given by

$$s_k^{ij'} = \begin{cases} s_k^{ij} & , i, j \notin I_k^1 \\ 0 & , \text{otherwise} \end{cases} \quad (24)$$

depending on the elements s_k^{ij} of \mathbf{S}_k . With δ_i we denote the i -th component of δ , the same holds for \underline{x}_i and \bar{x}_i , respectively. The set $I_k(\delta)$ contains the indices of blocked directions, i.e., x_i is located on a constraint boundary and δ_i points towards the constraint. Constraints contained in I_k^1 block movements along the directions fulfilling the Karush-Kuhn-Tucker (KKT) conditions and I_k^2 blocks movements, which would leave the feasible region due to the linear transform described by \mathbf{S}_k . Finally given I_k the search direction is obtained via

$$\mathbf{d}_k = -\mathbf{S}_k'' \nabla f(\mathbf{x}_k) \quad (25)$$

and

$$s_k^{ij''} = \begin{cases} s_k^{ij} & , i, j \notin I_k \\ 0 & , \text{otherwise} \end{cases}. \quad (26)$$

During the optimization of the parameters of a single model, we compute the gradient $\nabla f(\mathbf{x})$ and $\mathbf{S}_k = -\nabla^2 f(\mathbf{x})^{-1}$ directly. When carrying out the experiments for the ensemble model this turned out to be too expensive computationally to be practical for our purposes. Instead of computing \mathbf{S}_k we used the Broyden-Fletcher-Goldfarb-Shanno (BFGS) approximation in conjunction with the Sherman-Morrison formula [23], [25] resulting in a Quasi-Newton step.

C. Line search

According to Fig. 3 an estimation of the step length is the next step in the optimization procedure. A step along \mathbf{d}_k can still leave the feasible region, when stepping too far. There is an upper limit $\bar{\alpha}_k$ of α imposed by the non-binding constraints

$$\bar{\alpha}_k = \min \left(\{1\} \cup \left\{ \frac{z_k^i - x_k^i}{d_k^i} \mid i \notin I_k \right\} \right), \quad (27)$$

$$z_k^i = \begin{cases} \underline{x}_i & , d_k^i < 0 \\ \bar{x}_i & , d_k^i > 0 \end{cases}. \quad (28)$$

In the case of an approximation of \mathbf{S}_k the line search was carried out using quadratic interpolation. The derivative information of

$$\phi_k(\alpha) = f(\mathbf{x}_k + \alpha \mathbf{d}_k) \quad (29)$$

is already available at $\alpha = 0$. Due to the calculation of $\nabla f(\mathbf{x}_k)$ and \mathbf{d}_k we get

$$\phi'_k(0) = \mathbf{d}_k^T \nabla f(\mathbf{x}_k). \quad (30)$$

Now a value $\beta \in (0, \bar{\alpha}_k]$ fulfilling $\phi_k(\beta) \geq \phi_k(0)$ is located. The minimum of the interpolation polynomial is given by

$$\gamma = \frac{1}{2} \frac{\phi'_k(0)\beta^2}{\phi'_k(0)\beta - (\phi_k(\beta) - \phi_k(0))}. \quad (31)$$

If ϕ_k is decreased sufficiently, i.e.,

$$\phi_k(\gamma) \leq \phi_k(0) + c\gamma\phi'_k(0), \quad (32)$$

where $c = 10^{-5}$, we set $\alpha_k = \gamma$ and the line search is finished. Otherwise β is replaced with γ and the process is repeated.

D. Stopping condition

The optimization algorithm stops, when all components $\nabla_i f(\mathbf{x}_k)$, $i \notin I_k$ are in the range $[-T, T]$. It turned out that the precision requirements are rather relaxed, $T \in [10^{-3}, 10^{-2}]$ gives satisfying results. A higher request in precision translates into compression gains typically below 0.0001 bpc, which can be considered insignificant. The number of iterations has been limited to 50.

E. Derivatives

To perform the optimization process it is necessary to calculate the partial derivatives, since these form the gradient and the Hessian. For reasons of convenience we introduce

$$h(y, p) = -y \ln p - (1 - y) \ln(1 - p). \quad (33)$$

Note that here \ln denotes the natural logarithm. Using this convention (19) becomes

$$f(\mathbf{x}) = \frac{1}{n \ln 2} \sum_{k=1}^n h(y_k, p_k(\mathbf{x})) \quad (34)$$

and a partial derivative w.r.t. x_i , a component of \mathbf{x} , is

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \frac{1}{n \ln 2} \sum_{k=1}^n \frac{\partial h(y_k, p_k(\mathbf{x}))}{\partial x_i}. \quad (35)$$

Since $y \in \{0, 1\}$ we may write

$$\frac{\partial^n h(y, p)}{\partial p^n} = (n-1)! \left[-\frac{y}{p} + \frac{1-y}{1-p} \right]^n, \quad (36)$$

The first derivative

$$\frac{\partial h(y, p)}{\partial x_i} = \frac{\partial h(y, p)}{\partial p} \frac{\partial p}{\partial x_i} \quad (37)$$

and the second derivative

$$\frac{\partial^2 h(y, p)}{\partial x_i \partial x_j} = \frac{\partial h(y, p)}{\partial x_i} \frac{\partial h(y, p)}{\partial x_j} + \frac{\partial h(y, p)}{\partial p} \frac{\partial^2 p}{\partial x_i \partial x_j} \quad (38)$$

can easily be obtained.

Single model: First the optimization of a single model is examined, i.e., $\mathbf{x} = (\lambda, \varepsilon)^T$. We can restate (8) as

$$p_k(\mathbf{x}) = \varepsilon + (1 - 2\varepsilon)q_k(\lambda) \quad (39)$$

where

$$q_{k+1}(\lambda) = q_k + \frac{1}{T_{k+1}}(y_k - q_k), \quad (40)$$

in the case of M_1 or

$$q_{k+1}(\lambda) = q_k + (1 - \lambda)(y_k - q_k), \quad (41)$$

for M_2 , cf. (13) and (15). Thus the required partial derivatives of p_k can be expressed as

$$\frac{\partial p_k}{\partial \lambda} = (1 - 2\varepsilon) \frac{\partial q_k}{\partial \lambda}, \quad (42)$$

$$\frac{\partial p_k}{\partial \varepsilon} = 1 - 2q_k, \quad (43)$$

$$\frac{\partial^2 p_k}{\partial \lambda^2} = (1 - 2\varepsilon) \frac{\partial^2 q_k}{\partial \lambda^2}, \quad (44)$$

$$\frac{\partial^2 p_k}{\partial \varepsilon \partial \lambda} = \frac{\partial^2 p_k}{\partial \lambda \partial \varepsilon} = -2 \frac{\partial q_k}{\partial \lambda}. \quad (45)$$

Depending on the choice of the model, see Section II-B, the term q_k remains a function of λ (13), (15). Utilizing the iterative nature of (13) the expressions for the exact model (M_1) are given by

$$\begin{aligned} \frac{\partial q_{k+1}}{\partial \lambda} &= \frac{\partial q_k}{\partial \lambda} - \Delta q'_k, \\ \frac{\partial^2 q_{k+1}}{\partial \lambda^2} &= \frac{\partial^2 q_k}{\partial \lambda^2} + \\ &+ \frac{1}{T_{k+1}} \left[2 \frac{\partial T_{k+1}}{\partial \lambda} \Delta q'_k - \frac{\partial^2 T_{k+1}}{\partial \lambda^2} \Delta q_k + \frac{\partial^2 q_k}{\partial \lambda^2} \right] \end{aligned} \quad (46)$$

with the abbreviations

$$\Delta q_k = \frac{1}{T_{k+1}}(y_k - q_k), \quad (48)$$

$$\Delta q'_k = \frac{1}{T_{k+1}} \frac{\partial T_{k+1}}{\partial \lambda} \Delta q_k, \quad (49)$$

$$\frac{\partial T_{k+1}}{\partial \lambda} = \lambda \frac{\partial T_k}{\partial \lambda} + T_k, \quad (50)$$

$$\frac{\partial^2 T_{k+1}}{\partial \lambda^2} = \lambda \frac{\partial^2 T_k}{\partial \lambda^2} + 2 \frac{\partial T_k}{\partial \lambda}. \quad (51)$$

Exponential smoothing (M_2), (15), yields the following expressions:

$$\frac{\partial q_{k+1}}{\partial \lambda} = \lambda \frac{\partial q_k}{\partial \lambda} - (y_k - p_k), \quad (52)$$

$$\frac{\partial^2 q_{k+1}}{\partial \lambda^2} = \lambda \frac{\partial^2 q_k}{\partial \lambda^2} + 2 \frac{\partial q_k}{\partial \lambda}. \quad (53)$$

For the initial step, $k = 0$, all derivatives have been initialized to be zero.

TABLE I
COMPRESSION RATES (CALGARY CORPUS) IN BPC AND AVERAGE
CONTEXT HISTORY LENGTH L OF DIFFERENT ORDER CONTEXT HISTORIES
FOR THE LAPLACE, KRICHEVSKY-TROFIMOV AND THE DEVELOPED M_1
AND M_2 ESTIMATORS (SECTION II-B).

Order	L	LP	KT	M_1	M_2
0	14793.1	4.749	4.731	4.717	4.719
1	328.5	3.581	3.525	3.528	3.554
2	37.4	3.252	3.115	3.075	3.222
4	5.3	3.965	3.671	3.442	3.620
8	2.2	5.651	5.371	5.013	5.101

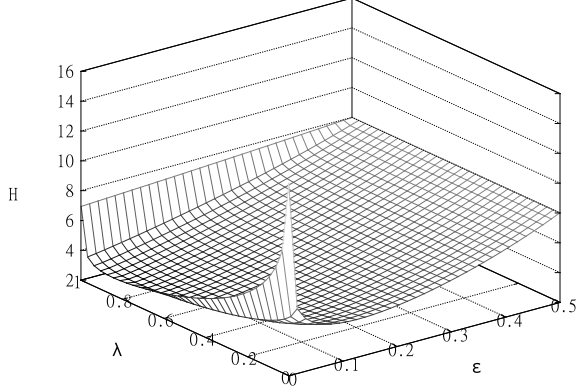


Fig. 4. Entropy $H(\lambda, \varepsilon)$ of the order-2 context histories of *bib*.

Ensemble model: As stated in Section II an ensemble model consists of an order-0 and an order-1 non-stationary model (predicting p_k^1 and p_k^2) and a switching probability, or weight ω . Thus a point in parameter space is $\mathbf{x} = (\lambda_1, \varepsilon_1, \lambda_2, \varepsilon_2, \omega)^T$. The expressions for calculating the gradient worked out above just need to be modified slightly. Higher order partial derivatives are estimated using BFGS. The partial derivatives of (33) are given by

$$\frac{\partial h(y, p_k)}{\partial z_i} = w_i \frac{\partial h(y, p_k)}{\partial p_k} \frac{\partial p_k^i}{\partial z_i}, \quad (54)$$

where $z_i \in \{\lambda_1, \varepsilon_1, \lambda_2, \varepsilon_2\}$, p_k is the mixed prediction in step k , see (17), and

$$w_i = \begin{cases} 1 - \omega & , i = 1 \\ \omega & , i = 2 \end{cases}. \quad (55)$$

Finally the remaining derivative for the ensemble model is

$$\frac{\partial h(y, p_k)}{\partial \omega} = \frac{\partial h(y, p_k)}{\partial p_k} (p_k^2 - p_k^1). \quad (56)$$

IV. EXPERIMENTAL EVALUATION

A. Single model

To evaluate a single prediction model we compare its compression performance against the well-known LP and KT estimators when forecasting conditional probabilities for different order- N context models. Note that N terms the number of source symbols, bytes in this case. The flat alphabet decomposition described in Section II-C has been applied. The

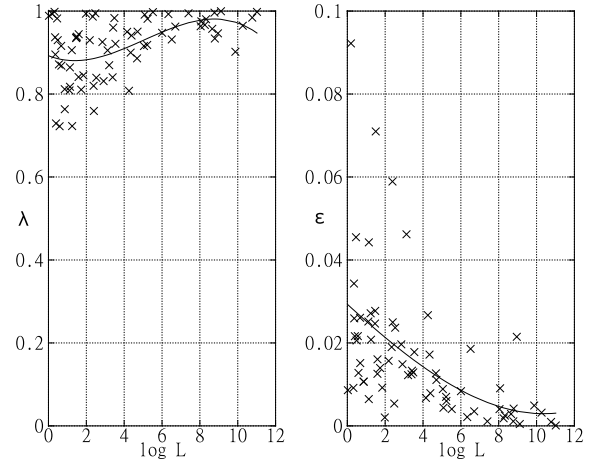


Fig. 5. Parameter values (λ, ε) and a third-order polynomial fit for M_1 , (13), as a function of average context history length L .

competing models are given by

$$p_k = \frac{S_k + \alpha}{T_k + 2\alpha}, \quad (57)$$

where S_k is the frequency of a one bit, T_k the total number of encountered bits in step k and α distinguishes the LP- ($\alpha = 1$) and KT-estimator ($\alpha = 0.5$). Whenever T_k reaches a threshold \bar{T} the frequencies T_k and S_k are halved (scaling). The parameter $\bar{T} \in \{1, 2, \dots, 1024\} \cup \{\infty\}$ was optimized for each file. Table I summarizes the average compression rates per context model for the Calgary Corpus and the average context history length. The estimator M_1 outperforms all other predictors, except in the case of an order-1 context, where its performance is slightly worse than a KT estimator. The LP estimator gives the worst overall results, probably due to the uniform prior-distribution assumed [2], [21]. Especially for short context histories [12], orders 2, 4 and 8, M_1 improves compression compared to the competing models. Exponential smoothing, M_2 , yields a good approximation when the context history contains at least a few hundred observations (order-0 and order-1).

Figure 4 depicts the typical shape of the cost function, (19), for M_1 . When λ is nearly zero the influence of past observations vanishes resulting in an unstable prediction behaviour, thus bad compression. A reasonable value of λ close to 1 gives good results. In the case of $\varepsilon \approx 0.5$ virtually no compression takes place, since the probability estimates fall within a narrow band around 0.5. A small value of ε near zero is a good choice. The estimator M_2 shows similar characteristics.

Figures 5 and 6 show the optimized values of λ and ε as a function of the context history length L . Shorter context histories seem to imply bigger values of ε on average. This resembles the observation made in [12], where it is stated that a bounded probability interval can show significant compression improvements for short sequences. The relation is more pronounced in the case of M_1 , see Fig. 5. The parameter λ grows as L decreases, again this effect is sharper when observing M_2 (Fig. 6). A possible explanation is the variable adjustment

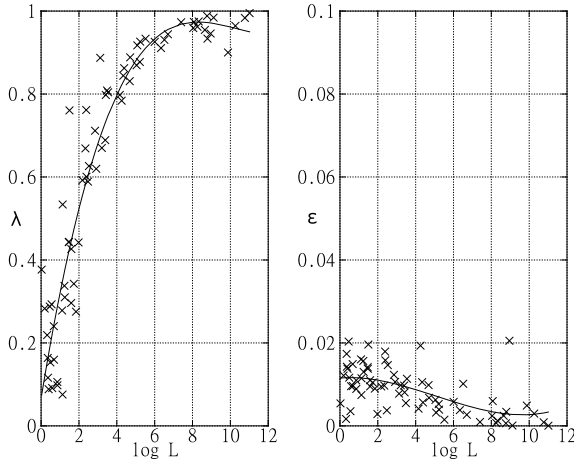


Fig. 6. Parameter values (λ, ϵ) and a third-order polynomial fit for M_2 , (15), as a function of average context history length L .

TABLE II

COMPRESSION RATES f_i (CALGARY CORPUS), NUMBER OF COST FUNCTION EVALUATIONS $\#f_i$ AND GRADIENT EVALUATIONS $\#\nabla f_i$ FOR M_i (SECTIONS II-B) USED IN AN ENSEMBLE MODEL, SEE SECTION II-D.

File	f_1 [bpc]	$\#f_1$	$\#\nabla f_1$	f_2 [bpc]	$\#f_2$	$\#\nabla f_2$
<i>bib</i>	1.945	16	10	1.959	5	3
<i>book1</i>	2.248	11	8	2.255	9	7
<i>book2</i>	1.955	9	6	1.962	4	3
<i>geo</i>	4.197	20	10	4.199	17	13
<i>news</i>	2.429	8	5	2.433	9	6
<i>obj1</i>	3.801	14	8	3.752	14	7
<i>obj2</i>	2.435	8	4	2.431	6	3
<i>paper1</i>	2.449	8	4	2.466	6	3
<i>paper2</i>	2.368	8	5	2.384	7	4
<i>pic</i>	0.712	35	19	0.727	14	11
<i>progc</i>	2.472	7	4	2.477	7	5
<i>progl</i>	1.703	9	7	1.721	8	7
<i>progp</i>	1.721	10	8	1.736	10	8
<i>trans</i>	1.521	11	9	1.528	11	9
Average	2.283	12.4	7.6	2.288	9.1	6.4

proportional to the prediction error. In M_1 the “adaption rate” $1/T_{k+1}$ (13) is large initially and decreases, opposed to M_2 where it is constant. Short sequences require a more rapid adaption, thus a constant “adaption rate” $1 - \lambda$ (15) should be high. We believe that the strong dependence of the parameters on L in the case of very short sequence (small L) is triggered by the small amount of observations rather than the actual statistical properties of a context history.

B. Ensemble model

In this section we simulate and compare the simple post-BWT-stage algorithm described at the end of Section II. Here we want to focus on the use of optimization in an online-scenario. After the BWT-output has been generated the model is optimized using different initial estimations

$$\mathbf{x}(M) = \begin{cases} (0.67, 0.002, 0.91, 0.005, 0.44)^T & , M = M_1 \\ (0.72, 0.003, 0.96, 0.004, 0.44)^T & , M = M_2 \end{cases} \quad (58)$$

For decompression the optimized parameters need to be transmitted along with the compressed data. Table II summarizes

the results. The estimator M_1 shows slightly better compression than M_2 on average, but requires more evaluations of the cost function and the gradient for optimization. A function evaluation directly corresponds to a compression pass, a gradient evaluation is slower, since more calculations are required. Oddly, the approximation M_2 outperforms M_1 on *obj1* and *obj2*. This indicates that the developed model does not fit the data characteristics in this particular case. The files *geo* and *pic* require many more iterations than the rest of the data, the optimal values of \mathbf{x} differ significantly from the rest of the corpus (and from the initial estimations), e.g.,

$$\mathbf{x}_{pic}(M) = \begin{cases} (0.922, 10^{-6}, 0.997, 0.002, 0.412)^T & , M = M_1 \\ (0.931, 10^{-6}, 0.951, 0.004, 0.297)^T & , M = M_2 \end{cases} \quad (59)$$

From a practical point of view M_2 achieves good compression, while requiring less resources during compression and offering faster model optimization. Finally Tab. III compares our best results to

- **BW94** [22] - the classical result of Burrows and Wheeler using Move-to-Front (MTF) and Huffman-Coding,
- **BS99** [26] - modified MTF and statistical modeling coupled with AC,
- **WM01** [13] - parsing and encoding (via AC) the BWT-output and omitting second-stage transformations and
- **D02** [27] - Weighted Frequency Counting (WFC), a different post-BWT transform, in conjunction with AC.

All of the other algorithms are rather complex, since they include either special post BWT transforms (e.g., MTF or WFC) and a statistical model or a sophisticated statistical model with a special parsing of the BWT output. Our approach is very simple and straight forward, since it just consists of a simple statistical model which processes the BWT output symbol by symbol (without a special parsing strategy). Taking the simplicity of our algorithm as a base it performs very well among the other approaches. The ensemble model gains 5% over BW94 and 1.3% over WM01. However, it compresses circa 1% worse than BS99 and 1.5% worse than D02. In the case of *book1*, *book2* and *pic* the ensemble model outperforms the other algorithms, showing the benefit of an optimized non-stationary model. The main drawback of the approach is that optimization is time consuming, since the online optimization requires to compress its input between 9 (M_2) and 12 (M_1) times on average (see Tab. II). But this can be neglected in a “distribution-scenario”: compression just takes place a few times and the compressed data is distributed, e.g., over the internet and needs to be decompressed multiple times.

V. CONCLUSION

In this paper a new approach to modelling non-stationary binary sequences was studied and possible low-complexity implementations have been shown. Using an iterative parameter-optimization method the parameters of the model can be fitted to training data automatically. In all test cases the new model shows a good performance compared to the LP- and KT-estimators. Both classic estimators are surpassed except in

TABLE III
COMPRESSION IN BPC OF VARIOUS BWT-BASED ALGORITHMS AGAINST
OUR BEST RESULT.

File	BW94	BS99	WM01	D02	best
<i>bib</i>	2.020	1.910	1.951	1.896	1.945
<i>book1</i>	2.480	2.270	2.363	2.274	2.248
<i>book2</i>	2.100	1.960	2.013	1.958	1.955
<i>geo</i>	4.730	4.160	4.354	4.152	4.197
<i>news</i>	2.560	2.420	2.465	2.409	2.429
<i>obj1</i>	3.880	3.730	3.800	3.695	3.801
<i>obj2</i>	2.530	2.450	2.462	2.414	2.435
<i>paper1</i>	2.520	2.410	2.453	2.403	2.449
<i>paper2</i>	2.500	2.360	2.416	2.347	2.368
<i>pic</i>	0.790	0.720	0.768	0.717	0.712
<i>prog</i>	2.540	2.450	2.469	2.431	2.472
<i>progl</i>	1.750	1.680	1.678	1.670	1.703
<i>progp</i>	1.740	1.680	1.692	1.672	1.721
<i>trans</i>	1.520	1.460	1.484	1.452	1.521
<i>Average</i>	2.404	2.261	2.312	2.249	2.283

one case, where our models show slightly worse results. Thus in the case of compressing non-stationary data the presented models typically improves compression. Beside the usage as a binary predictor on its own an ensemble model based on two non-stationary submodels for compressing BWT output has been designed. An alphabet decomposition is required to map the n -ary alphabet to a binary sequence, so the binary predictor can be used. The ensemble model contains an optimization pass prior to the actual compression. Such a simple ensemble model, together with online-optimization, shows good compression performance. Note that the ensemble model is very simple and does not apply any parsing strategies or post BWT transforms – it directly models symbol probabilities of plain BWT output. In order to make such an approach more practical further steps need to be taken to speed up the optimization process. Combining multiple models in data compression is highly successful in practice, but more research in this area is needed.

ACKNOWLEDGMENT

The author would like to thank Martin Aumüller, Michael Rink and Martin Dietzfelbinger for helpful suggestion and corrections, which improved the readability and made this paper easier to understand.

REFERENCES

- [1] G. V. Cormack and R. N. Horspool, "Data Compression Using Dynamic Markov Modelling," *The Computer Journal*, vol. 30, pp. 541–550, 1986.
- [2] F. Willems, Y. M. Shtarkov, and T. J. Tjalkens, "The context-tree weighting method: basic properties," *IEEE Transactions on Information Theory*, vol. 41, pp. 653–664, 1995.
- [3] M. Mahoney, "Adaptive Weighing of Context Models for Lossless Data Compression," Florida Tech., Melbourne, USA, Tech. Rep., 2005.
- [4] D. Salomon and G. Motta, *Handbook of Data Compression*, 1st ed. Springer, 2010.
- [5] T. Bell, I. H. Witten, and J. G. Cleary, "Modeling for text compression," *ACM Computing Surveys*, vol. 21, pp. 557–591, 1989.
- [6] M. Kufleitner, E. Binder, and A. Fries, "Combining Models in Data Compression," in *Proc. Symposium on Information Theory in the Benelux*, vol. 30, 2009, pp. 135–142.
- [7] X. Wang and N. J. Davidson, "The Upper and Lower Bounds of the Prediction Accuracies of Ensemble Methods for Binary Classification," in *Proc. International Conference on Machine Learning and Applications*, vol. 9, 2010, pp. 373–378.

- [8] J. Wichard and M. Ogorzalek, "Time series prediction with ensemble models," in *Proc. IEEE International Joint Conference on Neural Networks*, vol. 2, 2004, pp. 1625–1630.
- [9] M. Filho, T. Ohishi, and R. Ballini, "Ensembles of Selected and Evolved Predictors using Genetic Algorithms for Time Series Prediction," in *Proc. IEEE Congress on Evolutionary Computation*, vol. 8, 2006, pp. 2872–2879.
- [10] P. G. Howard and J. S. Vitter, "Practical Implementations of Arithmetic Coding," Brown University, USA, Tech. Rep., 1991.
- [11] —, "Analysis of arithmetic coding for data compression," in *Proc. Data Compression Conference*, vol. 1, 1991, pp. 3–12.
- [12] G. Shamir, T. Tjalkens, and F. Willems, "Low-complexity sequential probability estimation and universal compression for binary sequences with constrained distributions," in *Proc. IEEE International Symposium on Information Theory*, vol. 21, 2008, pp. 995–999.
- [13] A. Wirth and A. Moffat, "Can we do without ranks in Burrows Wheeler transform compression?" in *Proc. Data Compression Conference*, vol. 11, 2001, pp. 419–428.
- [14] P. Skibinski and S. Grabowski, "Variable-length contexts for PPM," in *Proc. Data Compression Conference*, vol. 14, 2004, pp. 409–418.
- [15] P. A. J. Volf and F. M. J. Willems, "The switching method: elaborations," in *Proc. Symposium Information Theory in the Benelux*, vol. 19, 1998, pp. 13–20.
- [16] F. Ghido and I. Tabus, "Optimization-quantization for least squares estimates and its application for lossless audio compression," in *Proc. Acoustics, Speech and Signal Processing*, vol. 33, 2008, pp. 193–196.
- [17] M. Drinic and D. Kirovski, "PPMexe: PPM for compressing software," in *Proc. Data Compression Conference*, vol. 12, 2002, pp. 192–201.
- [18] Y. Zhang and D. A. Adjeroh, "Prediction by Partial Approximate Matching for Lossless Image Compression," *IEEE Transactions on Image Processing*, vol. 17, pp. 924–935, 2008.
- [19] P. Skibiński, "Reversible data transforms that improve effectiveness of universal lossless data compression," Ph.D. dissertation, University of Wrocław, 2006.
- [20] D. Shkarin, "PPM: one step to practicality," in *Proc. Data Compression Conference*, vol. 12, 2002, pp. 202–211.
- [21] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Wiley-Interscience, 2006.
- [22] M. Burrows and D. J. Wheeler, "A block-sorting lossless data compression algorithm," digital Systems Research Center, Paolo Alto, USA, Tech. Rep., 1994.
- [23] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Athena Scientific, 1999.
- [24] R. Schaefer, *Foundations of Global Genetic Optimization*, 1st ed. Springer Publishing, 2007.
- [25] D. Kim, S. Sra, and I. S. Dhillon, "Tackling Box-Constrained Optimization via a New Projected Quasi-Newton Approach," *SIAM Journal on Scientific Computing*, pp. 3548–3563, 2010.
- [26] B. Balkenhol and Y. M. Shtarkov, "One attempt of a compression algorithm using the BWT," Universität Bielefeld, Tech. Rep., 1999.
- [27] S. Deorowicz, "Second step algorithms in the Burrows-Wheeler compression algorithm," *Software Practice and Experience*, vol. 32, p. 2002, 2001.